

METHOD AND SYSTEM OF ROUTING NETWORK-BASED
DATA USING FRAME ADDRESS NOTIFICATION

Field of the Invention

This invention relates to controlling data
flow in a data communications network, and more
particularly, to the routing of data by reading address
5 fields in hierarchal data structures.

Background of the Invention

Data networks have become increasingly
important in day-to-day activities and business
10 applications. Most of these networks are a packet-
switched network, such as the Internet, which uses a
Transmission Control Protocol (TCP) and an Internet
Protocol (IP), frequently referred to as TCP/IP. The
Transmission Control Protocol manages the reliable
15 reception and transmission of network traffic, while
the Internet Protocol is responsible for routing to
ensure that packets are sent to a correct destination.

In a typical network, a mesh of transmission
links are provided, as well as switching nodes and end
20 nodes. End nodes typically ensure that any packet is
received and transmitted on the correct outgoing link
to reach its destination. The switching nodes are
typically referred to as packet switches, or routers,
or intermediate systems. The sources and destinations
25 in data traffic (the end nodes) can be referred to as
hosts and end systems. These hosts and end systems
typically are the personal computers, work stations and
other terminals.

To help move information between computers, the open system interconnection (OSI) model has been developed. Each problem of moving information between computers is represented by a layer in the model, and thus, establishes a framework for standards. Two systems communicate only between layers in a protocol stack. However, it is desirable to communicate with a pure layer in the other system, and to achieve such results, information is exchanged by means of protocol data units (PDUs), also known as packets. The PDUs include headers that contain control information, such as addresses, as well as data. At a source, each layer adds its own header, as is well known to those skilled in the art. The seven layers, starting at the physical layer, include: (1) physical; (2) data link; (3) network; (4) transport; (5) session; (6) presentation; and (7) application layers.

The network systems typically use routers that can determine optimum paths, by using routing algorithms. The routers also switch packets arriving at an input port to an output port based on the routing path for each packet. The routing algorithms (or routing protocols) are used to initialize and maintain routing tables that consist of entries that point to a next router to send a packet with a given destination address. Typically, fixed costs are assigned to each link in the network and the cost reflects link bandwidth and/or costs. The least cost paths can be determined by a router after it exchanges network topology and link cost information with other routers.

The two lower layers, the physical and data link layers, are typically governed by a standard for local area networks developed by the IEEE 802 Committee. The data link layer is typically divided into two sublayers, the logical link control (LLC) sublayer, which defines functions such as framing, flow control, error control and addressing. The LLC

protocol is a modification of the HDLC protocol. A medium access control (MAC) sublayer controls transmission access to a common medium.

High-level data link control (HDLC) is a
5 communications control procedure for checking the accuracy of data transfer operations between remote devices, in which data is transferred in units known as frames, and in which procedures exist for checking the sequence of frames, and for detecting errors due to
10 bits being lost or inverted during transfer operations. There are also functions which control the set-up and termination of the data link. In HDLC, the bit synchronous data communication across a transmission link is controlled. HDLC is included in the ITU
15 packet-switching interface standard known as X.25.

Programmable HDLC protocol controllers are commonly used in these systems. An HDLC controller is a computer peripheral-interface device which supports the International Standards Organization (ISO) high-
20 level-data-link-control (HDLC). It reduces the central processing unit or microprocessor unit (MPU) software by supporting a frame-level instruction set and by hardware implementation of the low-level tasks associated with frame assembly-disassembly and data
25 integrity.

Most communication protocols are bit-oriented, code-dependent, and ideal for full duplex communication. Some common applications include terminal-to-terminal, terminal-to-MPU, MPU-to-MPU,
30 satellite communication, packet switching, and other high-speed data links.

A communication controller relieves a central MPU of many of the tasks associated with constructing and receiving frames. A frame (sometimes referred to
35 as a packet) is a single communication element which can be used for both link-control and data-transfer purposes.

2025 RELEASE UNDER E.O. 14176

[illegible]

A competing scheme is known as "cut through" (C/T). In this type of system, fragments of a packet or frame are moved into the external host memory, i.e., the shared system memory. The host processor is able to look at that portion of data, e.g., frame, at any time. For example, the processor looks at what exit port the packet should be moved, and initiates an

[illegible]

Overall, this was thought to be advantageous because the host processor would not have to wait until the end of the frame came in, but may still have part of the frame at the sending station. A problem, however, is that any problem on the wire could cause a malfunctioning or corrupt frame to occur. Thus, corrupt or bad frames would be propagated on the network.

Summary of the Invention

It is therefore an object of the present invention to provide a hybrid option between a store and forward architecture and a cut through
25 architecture.

It is still another object of the present invention to provide a method and apparatus of routing network-based data that improves over a cut through system where frames are moved from received ports to transport ports in constant bursts.

The present invention now allows an advantage over both store and forward (SF) architecture and cut through (C/T) architecture. In accordance with the present invention, the method routes the data frame and
35 comprises the steps of receiving at least a first portion of a frame within a FIFO receive memory of a network device. The first portion of the received

1. **Introduction**
 2. **Background**
 3. **Methodology**
 4. **Results**
 5. **Discussion**
 6. **Conclusion**
 7. **References**
 8. **Appendix**
 9. **Notes**
 10. **Tables**
 11. **Figures**
 12. **Supplementary Materials**
 13. **Author Contributions**
 14. **Conflicts of Interest**
 15. **Acknowledgments**
 16. **References**
 17. **Appendix**
 18. **Notes**
 19. **Tables**
 20. **Figures**
 21. **Supplementary Materials**
 22. **Author Contributions**
 23. **Conflicts of Interest**
 24. **Acknowledgments**
 25. **References**
 26. **Appendix**
 27. **Notes**
 28. **Tables**
 29. **Figures**
 30. **Supplementary Materials**
 31. **Author Contributions**
 32. **Conflicts of Interest**
 33. **Acknowledgments**
 34. **References**
 35. **Appendix**
 36. **Notes**
 37. **Tables**
 38. **Figures**
 39. **Supplementary Materials**
 40. **Author Contributions**
 41. **Conflicts of Interest**
 42. **Acknowledgments**
 43. **References**
 44. **Appendix**
 45. **Notes**
 46. **Tables**
 47. **Figures**
 48. **Supplementary Materials**
 49. **Author Contributions**
 50. **Conflicts of Interest**
 51. **Acknowledgments**
 52. **References**
 53. **Appendix**
 54. **Notes**
 55. **Tables**
 56. **Figures**
 57. **Supplementary Materials**
 58. **Author Contributions**
 59. **Conflicts of Interest**
 60. **Acknowledgments**
 61. **References**
 62. **Appendix**
 63. **Notes**
 64. **Tables**
 65. **Figures**
 66. **Supplementary Materials**
 67. **Author Contributions**
 68. **Conflicts of Interest**
 69. **Acknowledgments**
 70. **References**
 71. **Appendix**
 72. **Notes**
 73. **Tables**
 74. **Figures**
 75. **Supplementary Materials**
 76. **Author Contributions**
 77. **Conflicts of Interest**
 78. **Acknowledgments**
 79. **References**
 80. **Appendix**
 81. **Notes**
 82. **Tables**
 83. **Figures**
 84. **Supplementary Materials**
 85. **Author Contributions**
 86. **Conflicts of Interest**
 87. **Acknowledgments**
 88. **References**
 89. **Appendix**
 90. **Notes**
 91. **Tables**
 92. **Figures**
 93. **Supplementary Materials**
 94. **Author Contributions**
 95. **Conflicts of Interest**
 96. **Acknowledgments**
 97. **References**
 98. **Appendix**
 99. **Notes**
 100. **Tables**
 101. **Figures**
 102. **Supplementary Materials**
 103. **Author Contributions**
 104. **Conflicts of Interest**
 105. **Acknowledgments**
 106. **References**
 107. **Appendix**
 108. **Notes**
 109. **Tables**
 110. **Figures**
 111. **Supplementary Materials**
 112. **Author Contributions**
 113. **Conflicts of Interest**
 114. **Acknowledgments**
 115. **References**
 116. **Appendix**
 117. **Notes**
 118. **Tables**
 119. **Figures**
 120. **Supplementary Materials**
 121. **Author Contributions**
 122. **Conflicts of Interest**
 123. **Acknowledgments**
 124. **References**
 125. **Appendix**
 126. **Notes**
 127. **Tables**
 128. **Figures**
 129. **Supplementary Materials**
 130. **Author Contributions**
 131. **Conflicts of Interest**
 132. **Acknowledgments**
 133. **References**
 134. **Appendix**
 135. **Notes**
 136. **Tables**
 137. **Figures**
 138. **Supplementary Materials**
 139. **Author Contributions**
 140. **Conflicts of Interest**
 141. **Acknowledgments**
 142. **References**
 143. **Appendix**
 144. **Notes**
 145. **Tables**
 146. **Figures**
 147. **Supplementary Materials**
 148. **Author Contributions**
 149. **Conflicts of Interest**
 150. **Acknowledgments**
 151. **References**
 152. **Appendix**
 153. **Notes**
 154. **Tables**
 155. **Figures**
 156. **Supplementary Materials**
 157. **Author Contributions**
 158. **Conflicts of Interest**
 159. **Acknowledgments**
 160. **References**
 161. **Appendix**
 162. **Notes**
 163. **Tables**
 164. **Figures**
 165. **Supplementary Materials**
 166. **Author Contributions**
 167. **Conflicts of Interest**
 168. **Acknowledgments**
 169. **References**
 170. **Appendix**
 171. **Notes**
 172. **Tables**
 173. **Figures**
 174. **Supplementary Materials**
 175. **Author Contributions**
 176. **Conflicts of Interest**
 177. **Acknowledgments**
 178. **References**
 179. **Appendix**
 180. **Notes**
 181. **Tables**
 182. **Figures**
 183. **Supplementary Materials**
 184. **Author Contributions**
 185. **Conflicts of Interest**
 186. **Acknowledgments**
 187. **References**
 188. **Appendix**
 189. **Notes**
 190. **Tables**
 191. **Figures**
 192. **Supplementary Materials**
 193. **Author Contributions**
 194. **Conflicts of Interest**
 195. **Acknowledgments**
 196. **References**
 197. **Appendix**
 198. **Notes**
 199. **Tables**
 200. **Figures**
 201. **Supplementary Materials**
 202. **Author Contributions**
 203. **Conflicts of Interest**
 204. **Acknowledgments**
 205. **References**
 206. **Appendix**
 207. **Notes**
 208. **Tables**
 209. **Figures**
 210. **Supplementary Materials**
 211. **Author Contributions**
 212. **Conflicts of Interest**
 213. **Acknowledgments**
 214. **References**
 215. **Appendix**
 216. **Notes**
 217. **Tables**
 218. **Figures**
 219. **Supplementary Materials**
 220. **Author Contributions**
 221. **Conflicts of Interest**
 222. **Acknowledgments**
 223. **References**
 224. **Appendix**
 225. **Notes**
 226. **Tables**
 227. **Figures**
 228. **Supplementary Materials**
 229. **Author Contributions**
 230. **Conflicts of Interest**
 231. **Acknowledgments**
 232. **References**
 233. **Appendix**
 234. **Notes**
 235. **Tables**
 236. **Figures**
 237. **Supplementary Materials**
 238. **Author Contributions**
 239. **Conflicts of Interest**
 240. **Acknowledgments**
 241. **References**
 242. **Appendix**
 243. **Notes**
 244. **Tables**
 245. **Figures**
 246. **Supplementary Materials**

The amount of data to be transferred can be selected from the FIFO receive memory based on the desired address fields to be analyzed by the host processor. The method can also include the step of receiving the balance of the frame completely within the shared system memory. Also, an end-of-frame interrupt can be generated when a frame has been completely received within the shared system memory. A start-of-packet interrupt can be generated to a communications processor within the network device when the data received within the FIFO receive memory has reached a watermark value.

30 Additionally, a command can be issued to the direct memory access unit of the network device to transfer data from the FIFO receive memory to the shared system memory after the communications processor has received the start-of-packet interrupt. The method can further comprise the step of arbitrating the use of
35 a system bus between the direct memory access unit and a bus arbitration unit. Frame routing can then be

[illegible]

An interrupt bus can be connected between the FIFO receive memory and the communications processor. The HDLC port having the FIFO receive memory includes an interrupt generator for generating an interrupt to the communications processor along the bus. A FIFO bus can be connected between the direct memory access unit and the FIFO receive memory in which data is transferred from the FIFO receive memory and through the direct memory access unit to the shared system memory. The controller bus can be connected between the communications processor and the direct memory access unit through which data transfer commands are issued from the communications processor to the direct memory access unit. The FIFO receive memory has a watermark setting at which the HDLC port issues a start-up-packet interrupt for the communications processor.

[illegible]

5

10

15

20

25

t

30

a.

35

0000 **0000** **0000** **0000**

5

10

10

15

15

20

25

30

30

35

FIG. 25 illustrates a graph explaining how a watermark value has an inverse effect on the total number of generated interrupts.

	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415	2416	2417	2418	2419	2420	2421	2422	2
--	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	---

5 diagram of how the first-in/first-out memory overflows
on a second packet into a receive FIFO memory and the
various read and write status pointers.

10 and shared memory, and basic components of the network controller of the present invention, and showing the process when an early congestion notification signal is used for three different incoming packets with an overflow on the third packet.

15 FIG. 44 is a graph showing in detail
estimated traffic composition of the host bus with the
use of regular descriptors and the "fence-posting,"
when only the first and last descriptors are updated.

20 signaling between the host system and network device,
e.g., the network controller, of the present invention.

FIG. 46 is a flow chart describing the process of building descriptors within the network device.

25 FIGS. 47-50 are tables showing the various
fields of the receive and transmit message descriptors.

Detailed Description of the Preferred Embodiments

30 more fully hereinafter with reference to the accompanying drawings, in which preferred embodiments of the invention are shown. This invention may, however, be embodied in many different forms and should not be construed as limited to the embodiments set
35 forth herein. Rather, these embodiments are provided so that this disclosure will be thorough and complete, and will fully convey the scope of the invention to

those skilled in the art. Like numbers refer to like elements throughout.

Referring now to FIGS. 1-3, and more particularly to FIGS. 1 and 2, there is illustrated a high level diagram of a network controller and host system that are exemplary of the present invention. The network controller is an HDLC controller in one specific embodiment of the invention.

The present invention can be used in a number of different networks, including a conventional network making use of network controllers. For example, the invention could be used in many local area networks where computers are connected by a cable that runs from interface card to interface card. A wiring hub could provide a central point for cables attached to each network interface card. Hubs could connect connectors such as coaxial, fiber optic and twisted pair wire. One type of configuration could use unshielded twisted pair wire known as ten base T because it uses 10 megabits per second (NBPS) signaling speed, direct current, or base band, signaling and twisted pair wire.

The network could typically include routers, such as those that examine destination addresses contained in Net Ware IPX protocol. The routers would strip off the Internet packet, ring frame or other information and could send an IPX packet and any of its encapsulated data across a link. Any bridges could examine the address of each Internet packet and sent it across the circuit.

FIG. 1 illustrates a typical high level system diagram, which is illustrative of the general method, apparatus and system of the present invention. As illustrated, four network controllers 40, also known as network devices, connect into a 32-bit system bus 42, which is connected to host system 43. A host microprocessor 44 connects to the system bus 42, as does the shared memory subsystem 46. Each controller

[illegible]

5 performance, four port, high speed network controller designed for use in next generation bridge and router equipment, as well as any equipment requiring HDLC operation at T3 speeds. Each network controller is preferably manufactured as a single chip.

As shown in FIG. 2, on the network side, the network controller 40 contains four ports as noted before, numbered 0 to 3, 50, 52, 54 and 56, each with separate transmit and receive FIFOs allowing half or full duplex operation. Each port 50-56 has a transmit data handler 60 that receives transmit clock signals (TCLK) and forwards data signals (T Data) to line transceivers 62. The receive data handler 64 also receives clock signals (RCLK) and sends data to and from the line transceivers 62. The ports also each include the illustrated transmit and receive First-In/First-Out (FIFO) logic circuits 66,68; the 512 byte transmit FIFO 70, control circuit 74, and the 512 byte receive FIFO 72. The 512 byte FIFOs 70,72 connect to the frame bus 76 and the control circuit 74 connects to the management bus 78. The FIFO logic circuits 66,68, and data handler 60,64 and the control 74 work as appropriate transmit and receive circuitry for the transmit and receive (Tx), (Rx) 512 byte FIFOs.

On the system side, the controller 40 has a high speed (from 25 to 33 MHz), 32-bit system bus interface control unit (SBI) 80 which uses single cycle word transfers to minimize the controller's system bus usage and maximize its performance. The direct memory access unit (DMA) operation enables the device to

[illegible]

```
20  interrupt handler 104.
```

those skilled in the art.

bus 42. In this operation, the controller 40

5 controller's on-chip configuration/status registers by
using the same bus operating in a bus slave mode.

10 A four stage pipelined control unit is used to effectively execute one instruction per clock cycle, as typical. To provide the high performance required by this architecture, the internal SRAM 100 used by the communications processor could have three ports, and is
15 typically referred to as a Tri-Port RAM (TPR). The use of this architecture could allow a read from one register (or TPR), an ALU operation, and a write to a different register or TPR location, to all occur within the same clock cycle with one instruction.

The network controller 40 uses a phase locked loop (PLL) to generate an internal system clock from the externally provided system clock. This PLL generated system clock is delayed in time so as to minimize the signal to system clock delays which can impact performance. Consequently, the controller system clock must be 25 or 33 MHZ.

For purposes of understanding, a broad overview of operation is given, while referring to 35 FIGS. 1-8, followed by greater details of operation with reference to subsequent drawings. Once the

controller has been initialized and the ports are up and running, a typical frame reception proceeds as follows. The binary 01111110 pattern of the opening flag of the frame is detected by the HDLC port receiver
5 circuitry, which includes the Rx FIFO logic 68, Rx data handler 64 and line transceivers 62. This serial, digital data stream flows to the HDLC port's receiver circuitry where a search for the start-of-frame (a non-flag pattern) is performed to establish the octet
10 alignment and beginning of the frame. Frame check sequence (FCS) calculation begins on the first octet after the actual frame.

A serial to 32-bit parallel word conversion is performed by the receiver circuitry and the data
15 words are stored in the receiver (Rx) FIFO 74. Assuming the Rx FIFO 74 was empty at the start of this scenario, receive data continues to fill the receive FIFO 74 until the number of words therein is greater than the programmed watermark setting. As will be
20 explained in greater detail below, at this point, an interrupt is issued to the firmware 102 running on the on-chip RISC 92 requesting a data transfer for the receive FIFO 74. This interrupt is internal to the network controller 40 and is not visible to the host
25 system 44.

Upon receipt of the interrupt, the firmware 102 checks its on-chip copy of a current receive descriptor (fetched previously) for the requesting port. If it does not have ownership of a buffer, it
30 will direct the on-chip DMA to refetch the appropriate descriptor for examination. The controller 40 will repeatedly fetch the descriptor until one of two events occur: (1) it is given ownership of the buffer, or (2) the receive FIFO overflows (the frame is lost in this
35 case). Once buffer ownership is granted, the firmware

responds to the interrupt by directing the DMA to transfer a burst-size of frame data words from the receive (Rx) FIFO 74 to a receive buffer in system memory. Upon transfer of the first burst of the
5 received frame to system memory, a FAN (Frame Address Notification) interrupt may then be generated to the host via a Master Interrupt Register (MIR).

A cycle of receive FIFO 74 filling (by the network controller receiver circuitry), receiver-to-
10 firmware interrupts, and FIFO emptying (by the DMA) continues until the end of the frame is encountered by the receiver circuitry. At this point, the frame check sequence (FCS) of the frame is checked by the receiver circuitry and a receive status word is generated and
15 appended behind the frame in the receive FIFO 74.

Receiver-to-firmware interrupts continue until the remainder of the frame and the receive status word have been transferred to the receive buffer in system memory, as explained below. The firmware uses the on-
20 chip DMA 85 to update ownership, message size, error flags, etc. in the receive descriptor and then issues a "Frame Received" interrupt (RINT) to the host via the Master Interrupt Register (MIR) (FIG. 8B) indicating a completed reception.

A typical frame transmission takes place as follows. All frames are transmitted by the network controller 40 from transmit frame data buffers 204 assigned to entries in a transmit descriptor ring 202 (FIG. 3). When the system is ready for the network
30 controller 40 to transmit a frame, it relinquishes ownership of the associated transmit descriptor(s) and then does one of two things: (1) waits for the controller's transmit poll timer to expire causing the chip to poll the Tx descriptor in search of a buffer it
35 owns, or (2) is issued a Transmit Demand (TDMD) via the System Mode Register (SMR) by the host. In either

98-C-020-5223150

[illegible]

A cycle of emptying (by the transmitter unit) and filled (by the DMA) continues until the end of frame (EOF) has been written into the FIFO. When the transmitter removes the last data of the frame from the transmit FIFO, it optionally appends the FCS it has calculated (FCS appending by controller can be controlled on a frame by frame basis). The transmitter closes the frame by sending a closing flag(s).

30 The embedded processor 92 inside the network
controller 40 maintains 12 statistics in registers
on-chip for the host system to use. These statistics
are accessed by the host using a bus-slave
configuration/status register operation. As an
35 additional feature, the controller can be requested to

[illegible]

performs three key functions in DMA mode: (1) DMA engine for HDLC frame data transfers (bus master); (2) microprocessor port for access to configuration/status registers (bus slave); (3) and source for preferably two interrupts pins (MINTR# and PEINTR#). Both bus master and bus slave operations utilize the same 32-bit data bus and share some of the same control signals. There would be separate pins to select a proper mode for bus slave operations (CBIG) and bus master operations (DBIG).

contains the multi-channel DMA unit 85 for performing block data transfers with system memory 46 via a shared bus 42 without the involvement of the host processor 44. The controller requests ownership of the system bus whenever it has need to access an administration block 200, a transmit or receive descriptor 206, or a transmit or receive frame data buffer 204, as will be explained below with reference to FIG. 3.

Each time the network controller 40 accesses one of these data structures, it negotiates for bus ownership, transfers data (this may be several words), and then relinquishes bus ownership. For a given bus ownership, only sequential addresses are accessed. The size of each bus transaction (the number of words transferred or "burst size") can vary and is programmable for frame data transfers and statistics dumps. Administration block 200 and descriptor transfer size is determined by the network controller 40 on an as-needed basis, and can range from one to thirty-two consecutive words. The DMA unit 85 inside the system bus interface unit 80 provides the necessary

5 bit data bus that is used for DMA transfers. For this reason, register accesses cannot be performed when the controller is the bus master. Configuration/status ("config" for short) operation is designed to work with most popular microprocessors. All locations inside the
10 network controller could be implemented as 32-bit registers. All configuration and status registers, along with all of the network statistics, could be accessed via this interface.

15 controller 40 of the present invention involves three important system memory data structures: (1) administration block 200; (2) descriptor rings 202 with descriptors 206; and (3) frame data buffers 204. For any given application, one administration block 200, 20 eight descriptor rings 202 (FIG. 3) and multiple frame data buffers 204 are used. There is one descriptor ring 202 for each FIFO 70,72 at each port as illustrated in FIG. 3. Before initializing the controller 40, the host 44 is expected to allocate and 25 configure these data structures in system memory. The administration block 200 is used for chip initialization and as an exchange point for network statistics maintained by the controller.

30 with entries or descriptors 206 containing pointers and
information for frame data buffers 204 as is known to
those skilled in the art. Examples of devices and
systems showing the use of descriptors and descriptor
rings are disclosed in U.S. Patent No. 5,299,313 and
35 5,136,582, the disclosures which are hereby

incorporated by reference. Each descriptor ring 202 is dedicated to a specific FIFO 70,72 within the controller 40 and each two-word descriptor entry 206 within a ring is associated with one specific frame data buffer 204 in system memory (FIG. 5). Data buffers are defined as blocks of memory (typically ranging from 512 to 2,048 bytes) containing frames for transmission or providing space for frame reception.

As part of the initialization of the controller 40, the host must set aside a section of system memory. This memory is used to hold buffer management pointers, configuration information and per port network statistics. Since the administration block 200 can be updated periodically with statistics and can be referenced by the controller 42, it must remain an active allocation of memory throughout the operation of the device.

The administration block 200 (also called initialization block) consists of 512, contiguous bytes, and is word-aligned in memory. FIG. 7 illustrates greater details of the administration block 200 and its details. The first 15 words 200a of the administration block contain information used for chip initialization. The controller always refers to an on-chip copy of this section unless instructed to fetch part or all from shared system memory 46 again. The initialization section 200a of the administration block 200 contains system memory pointers to the eight descriptor rings 202, and set up information for six on-chip timers and nine DMA bus master burst sizes (maximum number of words transferred for various types of data per bus ownership).

The next contiguous four words 200b can be used by the host 43 to define the geometry of the

2025 RELEASE UNDER E.O. 14176

descriptor rings 202 and associated frame data buffer dimensions in external shared memory 46, as will be explained below. The controller 40 can automatically construct the (transmit) TX and (receive) RX descriptor rings 202 (FIG. 3).

The remaining words 200c of the administration block 200 provide space for the controller 40 to copy images of its on-chip HDLC frame statistics into shared system memory 46 when instructed to do so by the appropriate primitive. These periodic statistics snapshots are for the system to use. Allocation of these words of the administration block 200 is not required if the statistics dump feature is not used.

After chip reset is complete, once a reset-in-progress pin has gone inactive, the initialization procedure can begin as shown in FIGS. 45 and 46, and explained in greater detail below with reference to Section V. First, the host sets up the admin block 200, descriptor rings 202 and frame data buffers 204 in system memory. Second, the host 44 writes the starting system address of the administration block 200 to a register inside the controller 40 called a "Pointer to the Administration Block" (PAB), and optionally enables primitive interrupts. Next, an interrupt (INT) primitive is issued by the host 44 to the network controller. This causes the controller to copy the first 32 words (FIG. 7) of the administration block 200 into the chip of the network controller for processing. The network controller then responds with an acknowledgment INIT_COMPLETE or ACK (INIT) primitive interrupt to the host. At this point, the host 44 is free to housekeep or configure all of the controller's registers, establishing modes of operation for each

[illegible]

The first eight entries in the administration block 200 are system addresses which act as pointers to the top of each descriptor ring 202 (FIG. 3). Since the descriptors 206 must be word-aligned (or byte-aligned) in memory, these pointers should always be programmed with zeros in the least significant two address bits (the byte address). In other words, all descriptor ring pointers should be evenly divisible by four. Unpredictable operation will results from non-aligned descriptor ring pointer addresses. The network controller 40 refers to its copy of these pointers once the INIT primitive is completed, changing the pointers in system memory after the INIT has no effect unless another INIT is performed or a refresh descriptor ring primitive is issued.

As noted before, each transmit channel and
each receive channel within each port 50, 52, 54 and 56
uses a dedicated descriptor ring 202 for a total of
eight rings (one transmit ring and one receive ring per
port) (FIGS. 3 and 4). A descriptor ring 202 (FIG. 4)
is a circular queue comprising of several two-word
entries called "descriptors 206". Each descriptor
entry 206 describes one frame data buffer 204. The
first word 208 of a descriptor 206 entry contains
information about its frame data buffer 204 and the
frame, or partial frame, that the frame data buffer
contains (FIG. 5). The second word 210 of a descriptor

[illegible]

For frame reception on any given port, the host 44 is required to provide the controller 40 with ownership of contiguous descriptors pointing to empty frame data buffers 204. After the very first words of the frame have been transferred to memory 46, a Frame Address Notification (FAN) interrupt is issued (FIGS. 13-21 explained below in greater detail in Section I). Once a frame is fully received by the controller, ownership of its constituent descriptors is then reassigned. The host is signaled regarding this event

[illegible]

For frame transmissions on a given port, the network controller 40 "follows" the host 44 around a transmit descriptor ring 202 leaving used buffer descriptors in its wake for the host to reclaim. The host only gives the controller 40 ownership of descriptors 206 when it has one or more frames ready for transmission. Once a frame is fully transmitted by the controller, ownership of its constituent descriptors 206 is passed back to the host 44 for reuse. The host 44 is signaled regarding this event via a TINT interrupt.

In some applications, the host 44 may elect to use frame data buffers 206 which are smaller in size than the frames received or transmitted. A single frame spans multiple buffers. This allows frames to be

[illegible]

20 contents of chained buffers. Transmission of the frame
terminates only when it encounters a buffer whose
descriptor has set the end-of-frame flag.

25 are chained together by updating the first and last
descriptor entries involved (FIG. 4). When the network
controller 40 is finished with the buffers involved in
a chained frame, it first returns ownership of the last
descriptor and then it returns ownership of the first
30 descriptor. These are the "fence posts" of the frame
(FIG. 44 and Section IV below). The host 44 assumes
ownership of all the intervening frame data buffers
even though they are owned by the controller. Hence,
whenever the host encounters a host-owned descriptor
35 not marked by the end-of-frame flag, it should assume

25449-25459
 25460-25470
 25471-25481
 25482-25492
 25493-25503
 25504-25514
 25515-25525
 25526-25536
 25537-25547
 25548-25558
 25559-25569
 25570-25580
 25581-25591
 25592-25602
 25603-25613
 25614-25624
 25625-25635
 25636-25646
 25647-25657
 25658-25668
 25669-25679
 25680-25690
 25691-25701
 25702-25712
 25713-25723
 25724-25734
 25735-25745
 25746-25756
 25757-25767
 25768-25778
 25779-25789
 25790-25800
 25801-25811
 25812-25822
 25823-25833
 25834-25844
 25845-25855
 25856-25866
 25867-25877
 25878-25888
 25889-25899
 25900-25910
 25911-25921
 25922-25932
 25933-25943
 25944-25954
 25955-25965
 25966-25976
 25977-25987
 25988-25998
 25999-26009
 26010-26020
 26021-26031
 26032-26042
 26043-26053
 26054-26064
 26065-26075
 26076-26086
 26087-26097
 26098-26108
 26109-26119
 26120-26130
 26131-26141
 26142-26152
 26153-26163
 26164-26174
 26175-26185
 26186-26196
 26197-26207
 26208-26218
 26219-26229
 26230-26240
 26241-26251
 26252-26262
 26263-26273
 26274-26284
 26285-26295
 26296-26306
 26307-26317
 26318-26328
 26329-26339
 26340-26350
 26351-26361
 26362-26372
 26373-26383
 26384-26394
 26395-26405
 26406-26416
 26417-26427
 26428-26438
 26439-26449
 26450-26460
 26461-26471
 26472-26482
 26483-26493
 26494-26504
 26505-26515
 26516-26526
 26527-26537
 26538-26548
 26549-26559
 26560-26570
 26571-26581
 26582-26592
 26593-26603
 26604-26614
 26615-26625
 26626-26636
 26637-26647
 26648-26658
 26659-26669
 26670-26680
 26681-26691
 26692-26702
 26703-26713
 26714-26724
 26725-26735
 26736-26746
 26747-26757
 26758-26768
 26769-26779
 26780-26790
 26791-26801
 26802-26812
 26813-26823
 26824-26834
 26835-26845
 26846-26856
 26857-26867
 26868-26878
 26879-26889
 26890-26900
 26901-26911
 26912-26922
 26923-26933
 26934-26944
 26945-26955
 26956-26966
 26967-26977
 26978-26988
 26989-26999
 27000-27010
 27011-27021
 27022-27032
 27033-27043
 27044-27054
 27055-27065
 27066-27076
 27077-27087
 27088-27098
 27099-27109
 27110-27120
 27121-27131
 27132-27142
 27143-27153
 27154-27164
 27165-27175
 27176-27186
 27187-27197
 27198-27208
 27209-27219
 27220-27230
 27231-27241
 27242-27252
 27253-27263
 27264-27274
 27275-27285
 27286-27296
 27297-27307
 27308-27318
 27319-27329
 27330-27340
 27341-27351
 27352-27362
 27363-27373
 27374-27384
 27385-27395
 27396-27406
 27407-27417
 27418-27428
 27429-27439
 27440-27450
 27451-27461
 27462-27472
 27473-27483
 27484-27494
 27495-27505
 27506-27516
 27517-27527
 27528-27538
 27539-27549
 27550-27560
 27561-27571
 27572-27582
 27583-27593
 27594-27604
 27605-27615
 27616-27626
 27627-27637
 27638-27648
 27649-27659
 27660-27670
 27671-27681
 27682-27692
 27693-27703
 27704-27714
 27715-27725
 27726-27736
 27737-27747
 27748-27758
 27759-27769
 27770-27780
 27781-27791
 27792-27802
 27803-27813
 27814-27824
 27825-27835
 27836-27846
 27847-27857
 27858-27868
 27869-27879
 27880-27890
 27891-27901
 27902-27912
 27913-27923
 27924-27934
 27935-27945
 27946-27956
 27957-27967
 27968-27978
 27979-27989
 27990-28000
 28001-28011
 28012-28022
 28023-28033
 28034-28044
 28045-28055
 28056-28066
 28067-28077
 28078-28088
 28089-28099
 28100-28110
 28111-28121
 28122-28132
 28133-28143
 28144-28154
 28155-28165
 28166-28176
 28177-28187
 28188-28198
 28199-28209
 28210-28220
 28221-28231
 28232-28242
 28243-28253
 28254-28264

10 For receive frames, the message size 218 (MSIZE) field
of the first descriptor in the chain is updated with
the byte count of the entire frame, not simply the byte
count of the associated frame data buffer (since this
is equal to the buffer size). However, the message
15 size field 218 of the terminal descriptor will contain
only the actual number of bytes occupied by frame data
in its associated buffer. This allows the host to
easily locate the receive status word stored in the
first complete word following the frame data in the
20 buffer (note that the four bytes of the status word are
not included in the count stored in the MSIZE fields).

Although not required, best performance is achieved when frame data buffers 204 are word aligned in memory and large enough that chaining is not needed.

In a typical store-and-forward application,
35 the host maintains a "pool" of empty, unallocated frame

data buffers 204 in system memory. Assignment of a frame data buffer 204 to a receive descriptor 206 effectively removes it from this pool. Once a frame data buffer 204 has been filled, it is reassigned or
5 switched to one or more transmit descriptors. When transmission is finished, the frame data buffer 204 is returned to the pool for reuse and the cycle repeats.

The next two words in the administration block 200 after the descriptor ring pointers 200d
10 contain timer reload and control information 200e. The controller uses a hardware prescale timer 220 (FIG. 6) and divider 222 to divide down the UCLK frequency 224. A prescale timer reload value 226 is used to adjust the output frequency of the prescale timer.
15 Typically, a prescale reload value is selected to result in a 20 millisecond (50 Hz) prescale timer period through faster and slower periods are possible. The output of the prescale timer 226 is used as the base increment frequency for several secondary 8-bit
20 timers 228 maintained inside the network controller 40. These secondary timers can be: statistics dump timer 230, ports 0-3 transmit descriptor poll timers (four) 232-238. Each of the five, 8-bit timers has an associated reload value which is established in the
25 administration block 200. The following equation shows how to calculate prescale timer reload values.

$$\text{Prescale Reload} = 65.536 - \frac{T_{\text{prescale}}}{16 \times T_{\text{UCLK}}}$$

30

where T_{prescale} is the desired prescale timer period and T_{UCLK} is the system clock period.

35

004539E-002000

Table 1: Typical Prescale Timer Reload Values

	f_{UCLK} (MHZ)	T_{UCLK} (ns)	Decimal Reload Value (20 ms)	16-Bit Hex Reload Value (20 ms)
5	33	30	23.869	0x5D3D
	25	40	34.286	0x7EE6

The next equation shows how to calculate secondary
10 timer reload values:

$$\text{Secondary Reload} = 265 - \frac{T_{\text{secondary}}}{T_{\text{prescale}}}$$

15 where: $T_{\text{secondary}}$ is the desired secondary timer period and
 T_{prescale} is the prescale timer period.

Table 2: Typical Secondary Timer Reload Values

	T_{prescale} (ms)	$T_{\text{secondary}}$ (seconds)	Decimal Reload Value	8-Bit Hex Reload Value
	20	0.5	231	0xE7
	20	1.0	206	0xCE
25	20	2.0	156	0x9C
	20	5.0	6	0x06

Each of the secondary timers has a
corresponding enable control bit contained in the timer
enables field of the administration block (FIG. 7). A
30 "one" enables the timer; a "zero" disables the timer.
The following table shows the bit positions of each of
the five secondary timer enables. The controller
refers to its on-chip copy of the enables once INIT is
35 completed. Changing the enables in system memory has
no effect unless another INIT is performed or a
TIMER_ENABLE primitive is issued (0x0F). The prescale
timer is automatically disabled if none of the
secondary timers are enabled.

40

60620 = 506940

Table 3: Administration Block Timer Enable Field
(1 = enabled; 0 = disabled)

Bit:	7	6	5	4	3	2	1	0
5	Reserved			Stats Dump	Tx 3 Poll	Tx 2 Poll	Tx 1 Poll	Tx 0 Poll

The granularity of the prescale timer 220 permits a wide range of timer resolution. When selecting a prescale timer reload value 226, each prescale timer expiration consumes a small fraction of the controller's on-chip processing bandwidth. Selecting a very small prescale timer period (large reload value) can unintentionally hamper the controller's ability to service incoming and outgoing frames and thereby effecting the overall performance of the device. It is recommended that the prescale timer not be operated below a one millisecond period (FIG. 6).

When selecting secondary timer reload values for the transmit descriptor poll timers 232-238, two factors should be considered: (1) the half or full duplex operating mode of the port; and (2) the expected traffic on a given port, e.g., the percent of available bandwidth that is actually used. In general, the greater the traffic, the higher the poll frequency. Some systems may opt not to use transmit descriptor polling, and instead rely on the transmit demand (TD) bits in the system mode register (SMR) to initiate frame transmission.

The next two words 200f in the administration block 200, after the timing words 200e, relate to burst size (the four bytes located at PAB+40) (FIG. 7), and indicate the individual burst sizes for DMA transfer of data to the corresponding transmit ports. The next four bytes (PAB+44) determine the burst sizes for DMA

transfer of frames from the corresponding receive ports. The DMA 85 will always transfer data in a burst size determined by the values set in these fields, until the remaining data to be transferred is less than
5 the selected burst size. The controller refers to an on-chip copy of these values once the INIT primitive has been completed. Subsequent changes must be indicated via submission of the appropriate primitive command.

10 Setting the burst and frame buffer sizes equivalent will minimize the required number of bus transfers per frame and provide improved performance, if system constraints will permit large DMA bursts.

 A system clock period 200g is located in byte
15 #1 of PAB+48, should contain the value "0x28" if operating at 25 MHZ or "0x1E" if operating at the 33 MHZ system clock. The controller refers exclusively to the on-chip copy of this value once the INIT primitive has been completed, changing this value in system
20 memory after INIT has no effect unless another INIT is performed.

 "N1" is a 16-bit variable that is selectable by the host for the maximum frame size to be received. The N1 values for Ports #0 and #1 are located at PAB+52
25 200h and for Ports #2 and #3 are located at PAB+56 200i. The N1 value would typically be programmed by the host at initialization and could range anywhere between one byte to 64K bytes. Typically N1 is 2K bytes or less for most applications. Any received
30 frame that exceeds N1 will cause the "Frames Larger Than N1" statistic to be incremented for that port. The controller 40 refers to an on-chip copy of these values once the INIT primitive is completed, changing these values in system memory after INIT has no effect
35 unless another INIT is performed.

00163033-00000000

THE **WORLD** **OF** **THE** **FUTURE**

A primitive exchange protocol must be followed by both host and firmware for the primitive mechanism to work properly. The host must issue one and only one primitive at a time, waiting for the provider primitive acknowledgment before issuing another primitive. On the other side, the firmware will generate one and only one provider primitive for each host primitive issued.

Other registers not described in detail, such as a Master Interrupt Mask Register (MIMR) and a Port Error Interrupt Mask Register (PEIMR), allow the host to select which corresponding MIR and PEIR interrupt

events will actually generate an interrupt on various pins. These registers do not effect the setting of bits in the MIR and PEIR, they only mask the generation of host interrupts as a result of an interrupt bit
5 being sent.

I. Frame Address Notification (FAN)

Referring now to FIGS. 9-21, there are illustrated further details and drawings showing the
10 frame address notification (FAN) interrupt that allows a hybrid option between the classic store and forward (SF) architecture and the cut-through (C/T) architecture. In accordance with the present invention, the frame address notification (FAN) is an
15 interrupt signaled to the host processor 44 when all relevant address fields for a received frame currently reside in shared memory 46. The frame may then be processed by an address and look-up engine with the appropriate algorithm and look-up table 46c (FIG. 20)
20 and dispatched to the proper port and destination. This provides that the pipelining effect because routing is permitted to occur in parallel while the remainder of a frame could be incoming off the network wire.

25 Additionally, by the careful selection of the DMA 85 burst-size, any appropriate address field can be made available when the initial burst is read of the frame. The MAC-level headers, IP addresses, or even the TCP/UDP ports could be read into memory depending
30 upon the size of the burst. This facilitates L2-L3 or L4 frame switching applications.

FIGS. 9, 10, 11 and 12 illustrate how the TCP/UDP header is encapsulated in an IP data area and the IP header contained in a MAC data area. FIG. 9
35 gives a good indication of layering. The TCP/UDP data area 240 and TCP/UDP header 240a, IP data area 242,

[illegible]

5

10

25

39?

enters in the affirmative with a yes. At the same time, the frame is still being received within the FIFO 72. At this time, the DMA 85 transfers data from the FIFO 72 to the shared system memory 46 as shown in FIG.

5 18. The first burst of this DMA 85, as illustrated in FIG. 18, will then cause the CPC 92 to issue an interrupt signal known as the FAN or frame address notification event to the host processor 44 via the system bus 42, indicative that the preselected address
10 fields of the frame are present in the shared memory 46 (FIG. 19). The amount of the DMA burst size has been adjusted for the particular headers and addresses that will be looked at and for what layers.

As shown in FIG. 20, the host processor 44
15 then initiates the look up algorithm and determines how the packet and frame is to be addressed and transferred. This look up and FAN event can occur even when a frame is still being received within the frame receive buffer.

20 An end-of-frame (EOF) interrupt is issued when a frame has been completely received within the shared memory 46. Thus, this signifies when the host can transfer or finish the transfer process.

FIG. 21 illustrates a timing chart showing
25 the frame address notification (FAN) event. As shown at the top with the MAC layer, a start-of-packet shown as P1 is first issued followed by the firmware (FW) instruction to the DMA to build the start-of-packet command with the receiver. A continuation of packet
30 (COP) command is issued and then, as illustrated, the DMA transfers data. DMA also issues the frame address notification and then issues the end-of-packet (EOP). A similar circumstance occurs with the second packet known as P2 as shown at the top at the MAC layer.

II. Look-Ahead Watermark

Referring now to FIGS. 22-25, greater details of the look-ahead watermark used in the present invention is disclosed. The look-ahead watermark

5 (LAWM) functions as a synchronizing signal where the FIFO (first-in/first-out memory) memory, which includes the transmit and receive FIFO 70,72 provides a look-ahead watermark (LAWM) to indicate sufficient storage exists to receive one or more additional write bursts.

10 The transmission of frames can be expedited by this technique because it increases the bus and memory resource utilization while reducing the load on the communications processor 92.

The look-ahead watermark signal implies that

15 the FIFO can accommodate an additional DMA burst of the indicated quantity. The DMA burst size is not required to be the same size as the look-ahead watermark-mediated burst. The look-ahead watermark functions more as a "capacity-indicator" than as a conventional

20 transmit "level-sensitive" watermark mechanism. In another respect, the look-ahead watermark is a "top-down" capacity indicator versus a standard "bottom-up" watermark.

The look-ahead watermark has advantages and

25 aids the processing of data. It allows a reduction or elimination of FIFO underflow errors. It improves the utilization of the direct memory access unit. It also expedites frame transfer. It allows the earlier detection of a next frame for transmission. It

30 improves the utilization of expensive FIFO memories and reduce network inter-frame gap timing "delays". It also allows a reduction in the cycles per frame, i.e., microprocessor workload, and allows efficiency enhancement for both small and large frames. It is

35 transparent to the host system and reduces the CPU context switching.

The look-ahead watermark allows the device (firmware/hardware state machine) to "look" into the FIFO memory to determine if it can support additional bursts of data (of a known quantity) and hence

5 eliminate/reduce one or more CPU context switches per frame. A second DMA command can be enqueued with little additional overhead to move the next frame burst to the destination FIFO.

FIG. 22 illustrates a conventional FIFO flow-
10 control versus look-ahead watermark. The drawing is an abstract portrayal of the basic concept of a FIFO memory structure showing the system side and the network side. The transmit watermark is indicated at 260. The timing mechanism is shown on the bottom
15 horizontal line and shows time with the data burst indicated at point 1 for a data burst X, and look-ahead watermark data burst Y at points 2 and 3. A look-ahead watermark timeline illustrates the firmware look-ahead watermark check. In the conventional example, the FIFO
20 is empty (data = 0) and then the interrupt is generated and one data burst then fills the FIFO such that the current data is X. With the firmware look-ahead watermark check, the firmware submits a command to the DMA for data transfer to the FIFO and the second data
25 burst occurs as shown by the numeral 2 and the current data becomes X+Y. The firmware then checks the look-ahead watermark and a third data burst occurs as indicated by the numeral 3 such that the current data becomes X+2Y.

30 As shown in the flow chart at FIG. 23, starting at block 300, the method of the present invention for controlling data flow in a data-based network using the network controller of the present invention with a look-ahead watermark is illustrated.
35 At block 300, the DMA burst size is stored, as well as a look-ahead watermark burst size. The two burst sizes can be substantially the same or different. The

0016393400

channel is then enabled. The watermark interrupt is then generated to the DMA at block 302. At block 304, the firmware issues a data transfer command to the DMA. As part of this command, the firmware then requests the DMA to acknowledge via a request for end of command (REOC) when the task is completed: REOC=TRUE. At block 306, the DMA then arbitrates for the extension bus and then transfers data to the transmit FIFO. It signals via an EOC flag when it is finished.

10 A decision occurs at block 308 to determine if the DMA transfer is complete, which corresponds to EOC=TRUE. If the DMA transfer is not complete, then block 306 is repeated. If the DMA transfer is complete, the FIFO control logic determines the data
15 capacity at block 310. As illustrated, the FIFO control logic calculates the data capacity by subtracting the current data value held within the FIFO from the maximum value that can be held within the FIFO. That result is then divided by the look-ahead
20 watermark burst size to obtain the data capacity. As shown in block 312, if the data capacity is greater than or equal to 1, the look-ahead watermark value (such as a flag) is true. If the look-ahead watermark value is less than 1, then it is false. If the look-ahead watermark flag is true at block 314, then an
25 additional command is issued to the DMA at block 316, and the DMA transfers data to the transmit FIFO at block 318. If the look-ahead watermark is false, then the routine terminates.

30 FIGS. 24a and 24b illustrate first an interrupt-mediated frame transmission (FIG. 24a) and a look-ahead watermark-mediated frame transmission (FIG. 24b). These timing mechanisms show the advantages of the look-ahead watermark and aids in quantifying the
35 efficiency of the look-ahead watermark in terms of the clock cycles. The charts show the staggered delay of the interrupts, such as when they are issued and

00163000-00000

1. DMA initiates frame transmission via a start of packet interrupt signal (SOP).
2. Firmware (FW) enables the transmit channel, builds a command (two 32-bit words) and submits this command to the DMA for execution.
3. DMA decodes the command, arbitrates for the external bus, reads appropriate data from external shared memory and writes this into the appropriate transmit FIFO memory.
4. After the DMA transfer completes and if the transmit watermark is not exceeded, then a continuation of packet (COP) interrupt will be generated.
5. Once again the firmware constructs a command and issues it to the DMA for execution.
6. If the firmware has not disabled the COP interrupt and data in the FIFO has not exceeded the standard watermark, then another COP may be generated.
7. An "end of packet" (EOP) interrupt is generated once the terminal byte of the frame is clocked out of the FIFO onto the network.
8. Firmware checks whether another frame is ready for transmission (i.e., chained).
9. In the event that a chained frame exists, a DMA command is then constructed and issued.
10. The first burst of the second frame is fetched from external RAM and written into the transmit FIFO memory.
11. Another COP is issued once the write burst terminates and if the FIFO WM is not exceeded.

13. If the firmware has not disabled the COP interrupt and data in the FIFO has not exceeded the standard watermark, then another COP may be generated.

15. Firmware checks whether another frame is
10 ready for transmission (i.e., chained), and if this is
not the case, disables the transmit channel.

1. DMA initiates frame transmission via a
15 start of packet interrupt signal (SOP).

3. DMA decodes the command, arbitrates for
20 the external bus, reads appropriate data from external
shared memory and writes this into the appropriate
transmit FIFO memory. If the LAWM signal indicates
sufficient capacity exists within the FIFO for an
additional burst, then the firmware will submit a
25 second command to the DMA for execution.

30 5. An "end of packet" (EOP) interrupt may
be generated once the terminal byte of the frame is
clocked out of the FIFO onto the network.

35 7. In the event that a chained frame
exists, a DMA command is then constructed and issued.

Run	Time	Temp	Pressure	Flow	Conc	Yield	Quality
1	10:00	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
2	10:15	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
3	10:30	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
4	10:45	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
5	11:00	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
6	11:15	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
7	11:30	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
8	11:45	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
9	12:00	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
10	12:15	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
11	12:30	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
12	12:45	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
13	13:00	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
14	13:15	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
15	13:30	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
16	13:45	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
17	14:00	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
18	14:15	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
19	14:30	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
20	14:45	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
21	15:00	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
22	15:15	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
23	15:30	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
24	15:45	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
25	16:00	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
26	16:15	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
27	16:30	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
28	16:45	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
29	17:00	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
30	17:15	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
31	17:30	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
32	17:45	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
33	18:00	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
34	18:15	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
35	18:30	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
36	18:45	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
37	19:00	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
38	19:15	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
39	19:30	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
40	19:45	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
41	20:00	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
42	20:15	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
43	20:30	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
44	20:45	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
45	21:00	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
46	21:15	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
47	21:30	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good
48	21:45	100°C	1.0 atm	1.0 L/min	0.1 M	0.5 g	Good

```

11.  Firmware checks whether another frame is
15 ready for transmission (i.e., chained), and if this is
    not the case, disables the transmit channel.

```

FIG. 25 shows a graph, illustrating watermark effects on interrupt generation with regard to packet size. The graph plots the number of generated interrupts as a function of FIFO watermark size. It can be observed from the graph that with an increase in packet size, the number of required interrupts also tends to increase. Watermark values have an inverse effect on the total number of generated interrupts. More often than not, manipulation of the watermark alone is insufficient in tuning the performance of a device. With high variability of network packet sizes and contention for shared system resources, an additional mechanism is desired. The look-ahead watermark of the present invention is such a mechanism and as such can be readily observed to depress the curves in FIG. 25.

III. Early Congestion Notification

66000-52691-00

The present invention also uses an early congestion notification signal or interrupt (ECN) for advanced host notification of congestion in a corresponding port receiver, such as one of the receive FIFOs 70. The term "advanced" can be used because earlier received frames may still be stored in the FIFO ahead of the errored frame. There could be anywhere from zero to a score of frames waiting to be dispatched, depending on the relative sizes of the FIFO and the sizes of the frames. Hence, there is potentially a significant delay between when an early congestion notification (ECN) is first signaled and the errored frame is processed. Previously, the host 44 was not aware of this type of error until its processing circuitry worked its way through the preceding frames and examined the status word of each and every frame until it came to the ill-fated frame. Because the host processor 44 was not aware of the overflow problem, its processing behavior continued to proceed unmodified and, therefore, numerous exceeding frames continued to overflow the FIFO and were therefore lost. This, of course, created a much greater demand on the upper level software to retransmit frames and, thus, create bandwidth problems in the network. Instead of a single downstream node with a lost frame problem, the situation rapidly developed into one where many downstream nodes were forced to reclock their transmit windows, easily exacerbating the problem.

In accordance with the present invention, as shown in FIG. 26 flow chart, a method for controlling network data congestion in the receive FIFO memory includes the step of generating a status error indicator within a receive FIFO memory indicative of a frame overflow within the FIFO memory (block 340). An

2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460 2461 2462 2463 2464 2465 2466 2467 2468 2469 2470 2471 2472 2473 2474 2475 2476 2477 2478 2479 2480 2481 2482 2483 2484 2485 2486 2487 2488 2489 2490 2491 2492 2493 2494 2495 2496 2497 2498 2499 2500 2501 2502 2503 2504 2505 2506 2507 2508 2509 2510 2511 2512 2513 2514 2515 2516 2517 2518 2519 2520 2521 2522 2523 2524 2525 2526 2527 2528 2529 2530 2531 2532 2533 2534 2535 2536 2537 2538 2539 2540 2541 2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554 2555 2556 2557 2558 2559 2560 2561 2562 2563 2564 2565 2566 2567 2568 2569 2570 2571 2572 2573 2574 2575 2576 2577 2578 2579 2580 2581 2582 2583 2584 2585 2586 2587 2588 2589 2590 2591 2592 2593 2594 2595 2596 2597 2598 2599 2600 2601 2602 2603 2604 2605 2606 2607 2608 2609 2610 2611 2612 2613 2614 2615 2616 2617 2618 2619 2620 2621 2622 2623 2624 2625 2626 2627 2628 2629 2630 2631 2632 2633 2634 2635 2636 2637 2638 2639 2640 2641 2642 2643 2644 2645 2646 2647 2648 2649 2650 2651 2652 2653 2654 2655 2656 2657 2658 2659 2660 2661 2662 2663 2664 2665 2666 2667 2668 2669 2670 2671 2672 2673 2674 2675 2676 2677 2678 2679 2680 2681 2682 2683 2684 2685 2686 2687 2688 2689 2690 2691 2692 2693 2694 2695 2696 2697 2698 2699 2700 2701 2702 2703 2704 2705 2706 2707 2708 2709 2710 2711 2712 2713 2714 2715 2716 2717 2718 2719 2720 2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731 2732 2733 2734 2735 2736 2737 2738 2739 2740 2741 2742 2743 2744 2745 2746 2747 2748 2749 2750 2751 2752 2753 2754 2755 2756 2757 2758 2759 2760 2761 2762 2763 2764 2765 2766 2767 2768 2769 2770 2771 2772 2773 2774 2775 2776 2777 2778 2779 2780 2781 2782 2783 2784 2785 2786 2787 2788 2789 2790 2791 2792 2793 2794 2795 2796 2797 2798 2799 2800 2801 2802 2803 2804 2805 2806 2807 2808 2809 2810 2811 2812 2813 2814 2815 2816 2817 2818

FIGS. 27A-G show a high level block overview of the early congestion notification method of the present invention. FIG. 27A indicates that the receive FIFO is empty and the read (RD) and write (WR) pointers are the same at 0,0. Data then begins to come in and the read pointer is at zero and the write pointer is advancing, as indicated in FIG. 27B. As the packet is received, the status is written in as indicated by the Stat 1. A second frame or packet arrives (Data 2) and begins to overflow (FIGS. 27C and 27D). When the overflow condition occurs, a flip-flop is set for an error, thus an overflow bit is set (FIG. 27G). At this point, the early congestion notification (ECN) is sent out. The write pointer is reset to the beginning of packet to and frozen until the end of packet occurs, at which the time error status field of low packet is entered. The read of the status 1 by the DMA copies it into the receive status register at the host address. No request of the DMA for another data transfer will

[illegible]

As shown in FIG. 32, the DMA negotiates for ownership of the system bus 42 with the bus arbitration logic unit 47, while data continues to transfer into the receive FIFO memory 72. In FIG. 33, the DMA 85 transfers data from the receive FIFO 72 to the shared system memory 46. As shown in FIG. 34, a second packet or frame then enters the receive FIFO memory 72. FIGS. 35, 36 and 37 are similar to FIGS. 30, 31 and 32, except that access to the system bus 42 has been denied. At this time, a third packet (dark shading) is entering (FIG. 38) in with the second packet (diagonal line shading). In FIG. 39, the incoming frame overflows the receive FIFO memory 72 and the internal interrupt is sent to the communications processor 92 after an early congestion notification (ECN) bit has been set (FIG. 27G). In FIG. 41, the communications processor 92 then sets the ECN bits for the port in the

appropriate register block of the DMA 85. In FIG. 42, the DMA 85 signals the early congestion interrupt along the system bus 42 to the host processor 44 and the DMA 85 then transfers data from the receive FIFO 72 to the shared system memory 46, as shown in FIG. 43. The third frame is lost. However, the upper level software can then transmit the frame.

IV. Fence Posting

Reference should once again be placed in greater detail above concerning the discussion of descriptor rings 202 and descriptors 206, referring once again to FIGS. 3, 4, 5 and 7. In addition to the graph of FIG. 44, it is evident that the present method and apparatus controls the transfer of data arranged in frames between the host 44 and network controller 40. Both share the system memory 46. In accordance with the present invention, only the first and last descriptors 206 are updated within a descriptor "chain" to enhances bus utilization and grant ownership of first and last descriptors and any intermediate descriptors to the desired host or controller.

As noted before, the host 44 can elect to use frame data buffers 204 which are smaller in size than the frames that have been received or transmitted and, thus, a single frame data buffer could span multiple frame data buffers 204. This would allow frames to be dissected or assembled by the network controller 40. Naturally, as noted above, multiple frame data buffers 204 could hold the constituent pieces of the frame by "chaining" the associated descriptors 206 together and consecutive entries in the descriptor ring 202 with the end-of-frame flag set in the last descriptor of the chain. The respective frame data buffer of a

descriptor entry 206, which is owned but whose end-of-frame flag is not set, is considered to be part of a frame and not an entire frame. The controller 40 can chain descriptors 206 together one-by-one as it fills each successive frame data buffer 204. When the end of a frame is finally received and transferred to the external shared memory 46, the end-of-frame flag is set in the last descriptor of the descriptor chain (FIG. 4).

During transmission, the controller 40 is able to sequentially construct a single frame and the contents of "chained" frame data buffers 204, which are naturally pointed to by the "chained" descriptors 206. Transmission of the frame terminates only when it encounters a frame data buffer 204 whose descriptor 206 has set the end-of-frame flag. This great improvement in bus utilization is brought about by the present invention where instead of the prior art of successively updating each spanned descriptor 206, only the first and last descriptors are altered, such as by updating the ownership bit within the descriptor for network received frames. These first and last updated descriptors form the "fence-posts" of the chain.

All the flags and fields of the first and last descriptor in a "fence-posted" chain are updated in order to provide accurate information about a frame once it has been fully transmitted or received. For example, for received frames, the message size field 218 of the first descriptor in the chain is updated with the byte count of the entire frame, not simply the byte count of the associated buffer because this is equal to the buffer size.

As noted above, FIG. 4 illustrates the administration block 200 with the chip initialization section 200a, and the four ports with the statistics

image 200b-e. The descriptor ring 202 is shown with the various descriptors 206, that point to frame data buffers using addresses. A frame data buffer is shown at the right. FIG. 5 shows a frame data buffer 204 with a descriptor 26 as a two-word entry, with an ownership bit (OB) 212 and end-of-packet (EOP) 214. The buffer size 216 and message size 218 is contained in the one word 208, and the buffer address 219 in the other word 210. The graph in FIG. 44 illustrates in detail that the use of only the first and last descriptors as explained above creates a flat line to reduce traffic along the bus.

FIG. 3 also illustrates in detail how the administration block 200 has pointers 200d (FIG. 7) which directly points to the different transmit rings 202, having the descriptors 206 with the buffer information 206a, such as geometry, and buffer addresses 206b.

V. Creation of the Descriptor Rings

The present invention is advantageous because the network device now assumes responsibility for the creation of the data and buffer structures, such as the descriptor rings. The network device 40 constructs transmit and/or receive descriptor rings 202 (FIG. 3) in externally shared memory 46. In the present invention, support is provided for full-duplex channels. The parameters dictating the number of descriptors in either the transmit or receive descriptor rings 202 and their respective frame data buffer dimensions are communicated via a parameter block (or administration block).

This administration block 200 is exchanged between a host system 43 and network device 40 at

initialization (FIG. 45) via a communication primitive under host control. This administration block 200 is stored (or mapped) into numerous variable fields of the memory 46. As noted above, if the field values for the transmit descriptor ring size or receive descriptor ring size are non-zero, then construction can be initiated. Otherwise, in the event the fields are zero, the network device 40 will not build the associated descriptor rings 202. The network device 40 expects the host 43 to have already built the data and memory structures in the shared memory 46. The geometry or length of the descriptor ring 202 and the sizes of the associated frame data buffers 204 varies and the descriptor rings 202 often vary from 50 to 500 descriptors in length, while the frame data buffers 204 vary from about 256 bytes up to about 2,000 or 5,000 bytes. Frame data buffer size is selected based upon the maximum supported frame size of interfacing networks. The overall memory allocated per port 50-56 is in the two megabyte range.

The frame data buffer size has relatively little effect on the time required to actually build the descriptor rings 202. However, the descriptor ring size is the limiting factor for the construction time. A block-mode construction optimization technique is used to reduce the build time. Descriptors 206 can be built on-chip in blocks of two and transferred to external memory 46 via the direct memory access unit 85.

This block size is alterable and could be easily included within the parameter of blocks in the future. The method and network device of the present invention offers advantages to the market, including a reduced time required for host software development,

and a size reduction of a host code. There can be expedited testing and faster network device initialization. Also, the present invention expedites system implementation for application design engineers.

5 In accordance with the present invention, a block of memory within the shared memory 46 is allocated by the host system 43, which maps the administration block 200 having the descriptor ring parameters 200b as noted before (FIG. 7). These
10 parameters include the geometry of the descriptor ring 202 and descriptors 204 to be formed within the shared memory. FIG. 7 shows the administration block and indicates that at four addresses PAD+60 to PAD+72, the buffer size, the transmit ring size, and receive ring
15 size.

As shown in FIG. 45, the administration block 200 has the base pointer set up at point 0 on the chart. The host system 43 issues a primitive for initialization (INIT at point 1) to the network device.
20 At the same time, the host 44 writes into the network device 40 the base address of the administration block 200. The network device 40 then "fetches" or reads the administration block from the shared memory (point 2) and then sends an acknowledgment (ACK) back to the host
25 that the administration block is received. This administration block is processed, while the host system may conduct additional housekeeping (point 3) after receiving the acknowledgment.

As the administration block 200 is processed,
30 the network device 40 constructs corresponding descriptors as blocks of data that point to the frame data buffers to be formed within shared memory.

FIG. 46 shows in greater detail a flow chart that illustrates how the descriptors can be formed by
35 the network device. The host provides the pointers to

[illegible]

- The administration block is read by the
10 network device (block 402) and a descriptor header word
is built (block 404). The descriptor address words are
built (block 406) and the descriptor address updated
block 408). The buffer point address is also updated
(block 410) and then the descriptor block is read out
15 by the network device to the host RAM which is part of
the shared system memory (block 412).

- There are a number of assumptions, such as the use of contiguous descriptors, and an even count. Typically, the buffers are contiguous and of uniform size. If the buffer pointers are not provided, then the firmware 102 will start buffers at a two-word offset from the calculated termination of a descriptor ring. If the administration block descriptor parameter hexadecimal word is "0X00000000," then no associated descriptor rings 202 will be built. The administration block transfer is required prior to other configuration primitives because the block will overwrite the settings. All descriptor ring dimensions must be even values and the frame data buffer size can be a 0 or 1 or no descriptor ring 202 will be built. All buffer

[illegible]

Other disclosures that are related to the present invention are set forth in patent applications entitled, "METHOD AND SYSTEM OF CONTROLLING TRANSFER OF DATA BY UPDATING DESCRIPTORS IN DESCRIPTOR RINGS," "LOOK-AHEAD WATERMARK FOR ADDITIONAL DATA BURST INTO FIFO MEMORY," "METHOD AND APPARATUS FOR CONTROLLING NETWORK DATA CONGESTION," and "METHOD AND NETWORK DEVICE FOR CREATING BUFFER STRUCTURES IN SHARED MEMORY," which are filed on the same date and by the same assignee, the disclosures which are hereby incorporated by reference.

20 Many modifications and other embodiments of
the invention will come to the mind of one skilled in
the art having the benefit of the teachings presented
in the foregoing descriptions and the associated
drawings. Therefore, it is to be understood that the
25 invention is not to be limited to the specific
embodiments disclosed, and that the modifications and
embodiments are intended to be included within the
scope of the dependent claims.